## IN THE CLAIMS

1. (currently amended) A method for crawling a web site, the method comprising the steps of:

a) querying a web site server by a crawler program, wherein at least one page of the web site has a reference, wherein the reference is specified by a script ~~for executing by a browser~~ to produce an address for a next page;

b) parsing such a reference from one of the web pages by the crawler program and sending the reference to an applet running in a ~~the~~ browser; and

c) determining the address for the next page by the browser executing ~~responsive to the~~ reference and sending the address to the crawler.

2. (currently amended) The method of claim 1~~2~~, the browser being configured to use a certain proxy~~;~~ and refer to a resolver file for hostname-to-IP-address-resolution, ~~and~~ wherein the web site server has an IP address and ~~;~~ the proxy for the browser has a certain IP address, the certain IP address of the proxy being different than the IP address of the web site server, and wherein the resolver file indicates the certain IP address of the proxy as the IP address for the web site server.

3. (original) The method of claim 2, comprising the steps of:

adding an onload attribute to one of the web pages by the proxy;

defining an event handler for the onload attribute by the proxy, wherein the event handler sets a certain variable; and

polling the certain variable by the applet to determine when the page is loaded.

4. (currently amended) The method of claim 1, wherein the crawler is programmable to perform particular action sequences for selecting non-hypertext-link parameters from the at least one web page in a particular sequence, so that ~~generating~~ the queries to the web server include the selected parameters and a context arising from the particular sequence.

Najork describes the problem it addresses as follows:

> A web crawler is a program that automatically finds and downloads documents from host computers in an Intranet or the world wide web . . . Before the web crawler downloads the documents associated with the newly discovered URL's, the web crawler needs to find out whether these documents have already been downloaded . . . Thus, web crawlers need efficient data structures to keep track of downloaded documents and any discovered addresses of documents to be downloaded. Such data structures are needed to facilitate fast data checking and to avoid downloading a document multiple times.

Najork, col. 1, lines 34-61. Specifically, the teachings of Najork concern "the data structures and methods used to keep track of the URL's of documents that have already been downloaded or that have already been scheduled for downloading." Najork, col. 4, lines 54-57.

The web crawler taught by Najork includes "threads 130 for downloading web pages from the servers 112, and processing the downloaded web pages; a main web crawler procedure 140 executed by each of the threads 130; and a URL processing procedure 142 executed by each of the threads 130 to process the URL's identified in a downloaded web page." Najork, col. 3, lines 31-58. Each thread executes a main web crawler procedure 140 shown in FIG. 3. Najork, col. 4, lines 58-59. The web crawler thread determines the URL of the next document to be downloaded (step 160) and then downloads the document corresponding to the URL, and processes the document (162). Najork, col. 4, lines 59-64. According to that processing, the main procedure identifies URL's in the downloaded document that are candidates for downloading and processing (step 162). Najork, col. 4, line 66 - col. 5, line 3.

Najork specifically points out that "these URL's are typically found in hypertext links in the document being processed." Najork, col. 5, lines 3-4. But, as particularly pointed out in the present application, sometimes URL's are *not* found in hypertext links, which presents a problem. That is, one reason a conventional crawler and the crawler taught by Najork are not suitable for the "staticizing" problem addressed in the present invention is that "references from one web page to another may not be straightforward. That is, a reference may not be simply set out on the page as a hyperlink address [i.e., a URL], but instead may be a script, form, selection menu, or button for example. Thus a need exists for improvements in crawler programs, to overcome their limitations so that they may be used for the staticizing problem as well as other

applications." Present application, page 2, line 20 - page 3, line 6. Najork offers no teaching that addresses this problem, or even that suggests it exists.

The present application further elaborates on the problem, explaining how a reference that is not "simply set out on the page as a hyperlink address, but instead . . . specified by a script, for example, so that the address is produced only when a client browser executes the reference." Present application, page 5, lines 10 - 15; see also, page 12, line 21- page 13, line 1 (describing how the reference may be specified by a script, a selection menu, form, button or other element).

The present application goes on to explain how this problem may be addressed, as follows:

> To generate references of this sort in connection with generating the requests to the server, another aspect of the invention arises. According to an embodiment, the crawler parses each received web page and sends references to an applet developed for an embodiment of the present invention that runs in the browser. (This applet may be referred to herein as a "JavaScript execution engine" or simply "JEE.") The browser determines the address for a next page responsive to such a reference, so that the browser may receive the next page and any cookie for the next page from the server, and the JEE returns the address and any cookie to the crawler program.

Present application, page 5, lines 15-22; page 15, lines 2-8. Claims 1, 9 and 17 particularly point this out, as well.

Claim 1, as amended, states that ". . . a page has a reference, wherein the reference is specified by a script for producing an address for a next page." (Due to the introduction of "executing" in the last step of the claim, the language about "executing" is deleted from this first step.) The claim goes on to say that such a reference is parsed from one of the web pages by the crawler program and sent to an applet running in a browser. Further, the claim is herein amended to clearly tie the pieces together by stating that the address for the next page is determined by the browser "executing" the reference and sending the address to the crawler. Claims 9 and 17, as amended, have similar language.

No new matter is added in the amendments to claims 1, 9 and 17, since the specification as originally submitted provides support. Present application, page 5, lines 10-22 (regarding the reference having a script that is executed by a browser for the crawler).

The present application specifically states that according to the present invention "at least one page of the web site has a reference for *executing* by a browser to produce an address for a

next page." Present application, claims 1, 9 and 17 (emphasis added). Likewise, Applicant intended for this language to clearly point out something different than merely finding an address that is explicitly set out in a hypertext link, which is what is taught by Najork. Consider the following examples which further illustrate the difference.

An example of a hypertext link that explicitly has the text of an address set out therein, as alluded to by Najork, is as follows: **<a href="http://news.google.com/">**. In contrast, the following hyptertext link provides an example of a reference that is not so straightforward and that is "specified by a script to produce an address" so that the the address for the next page is determined "by the browser executing to the reference," as stated in amended claim 1 of the present application: **<a href="javascript:getMedia ('FA', '22-Sep-2004', '1', 'RM, WM');">**. See Web page, http://freshair.npr.org/day_fa.jhtml?display=day&todayDate=09/21/2004. Executing this reference produces a URL such as the following:

**http://www.npr.org/dmg/dmg.html?prgCode=FA&showDate=22-Sep-2004&segNum=1&NPRMediaPref=RM.**

The present application says "a reference . . . may be specified by a script" because an href tag, for example, typically has a *call* to a script and not the script *itself*. The browser locates the source code for the function that is called and then executes the specified function.

Note also, a "reference" is not limited to the context of an href tag. Consider the following example snippet of HTML code:

```
<form>
<input type="button" value="GO" onclick="DoSearch()"/>
<form>
```

This snippet creates a button that says "Go." When the user presses the button the browser needs to execute the function DoSearch() in the context of the button before it can determine what URL to load. In this example also the URL to be loaded is "specified by a script," the DoSearch() script, which is not itself included in the form that produces the button.

Note also the amended claim, like the passage of the specification set out above, states that the browser executes the *reference* instead of saying merely that the browser executes the script. In the snippet example above, the crawler needs to know what URL to load when the button is pushed. The crawler achieves this by telling the browser (via the applet) to push the button. It cannot tell the browser to just execute the JavaScript function "DoSearch()" because

the browser would then not have the context in which to execute the function. See present application, page 5, lines 10 - 15 (explaining that the address "is very dependent on the context in which it is produced, that is, the history that led up to it, including the state of the server and the client browser."); see also, page 15, lines 2-8 (explaining that the crawler passes information 230 to a JavaScript execution engine 210 for generating queries to the web server 100 and that the information includes the JavaScript command that invokes script 303 when button 304 is clicked, a context object, the browser window object, and the document object associated with page 140.X in its context as it exists, loaded in browser 205).

Applicant recognizes that *executing* a reference to produce an address, as claimed, might be confused with *parsing* the reference to find an address that is explicitly set out therein. The explanation above clarifies these significant differences. Also, to make the distinction particularly clear in the claims, Applicant herein submits the amendments described above to claims 1, 9 and 17.

It should be clear from the discussion above that the amended claims are patentably distinct from Najork, col. 4, line 59 - col. 5, line 4, which the Office action relies upon for the rejection. For these reasons Applicant contends that claims 1, 9 and 17 are allowable.

Claims 2, 10 and 18

The Office action relies upon Najork FIG. 1, domain name system 114 for the rejection of claims 10 and 18. Applicant assumes that claim 2 is likewise rejected on this basis, since claim 2, a method form of the invention, sets out a feature similar to what is set out in claims 10 and 18, which are different forms of the invention. Applicant herein amends claims 2, 10 and 18 , as set out above, to clearly distinguish the present invention over a conventional domain name server arrangement.

No new matter is added in the amendments to claims 2, 10 and 18, since the specification as originally submitted provides support for the amendments. Present application, page 6, lines 1-14.

As explained in the present application, due to the JavaScript execution engine ("JEE") used in an embodiment of the present invention to deal with JavaScript in hypertext references, a further difficulty arises because an applet running on a client browser can, for security reasons, only interact with objects in a web page if the web page and the applet are loaded onto the client

from the same server. Present application, page 6, lines 1-4; page 15, line 9 - page 16, line 2. To overcome these limitations, the client browser is configured to use a certain proxy gateway and to a certain file (referred to herein as the "resolver file") containing cross-referencing for hostname-to-IP-address-resolution. Present application, page 6, lines 6-9; page 15, lines 15-17. The resolver file indicates the IP address of the proxy gateway as the IP address for the web site server, even though the proxy gateway is not really the source of the web site server (hence the proxy gateway is referred to as a "spoof proxy"), so that the JEE running on the client *appears* to the client's browser to be from the same server as the web pages. Present application, page 6, lines 10-14; page 15, lines 17-23. This permits the JEE to communicate with the crawler and the browser unhindered by conventional limitations. Page 15, line 22 - page 16, line 2.

Consistent with the above, the present application specifically states that according to the present invention, "the browser [is] configured to use a certain proxy, and refer to a resolver file for hostname-to-IP-address-resolution, and wherein the web site server has an IP address, the proxy for the browser has a certain IP address, and the resolver file indicates the certain IP address as the IP address for the web site server." Present application, claims 2, 10 and 18 (emphasis added).

However, Applicant recognizes that the claims do not explicitly rule out the possibility that the proxy and the web site server have the same address. Therefore the arrangement described in the claims might be confused with a conventional DNS server arrangement, as described in the cited figure of Najork. Therefore, to make the distinction particularly clear, Applicant herein submits amendments to claims 2, 10 and 18 stating that "the certain IP address of the proxy [is] different than the IP address of the web site server." In a case like this, in which the proxy gateway is not really the source of the web site server, i.e., the gateway and the server have different IP addresses, it is not conventional for "the resolver file [to indicate] the certain IP address as the IP address for the web site server," as claimed. Clearly this is not taught or suggested by Najork or any of the other cited references.

For the above reasons Applicant contends that claims 2, 10 and 18 are allowable. Further, Applicant contends that the claims are patentably distinct because they respectively depend upon allowable claims.

Claims 4, 12 and 20

Claims 4, 12 and 20 state that the crawler is programmable to perform particular action sequences for generating the queries to the web server. For the rejection of these claims, the Office action relies upon Najork, col. 4, lines 59-62, which is discussed herein above in connection with claim 1. That is, the Office action equates the "action sequences" of the claims in the present application to the teaching by Najork about a web crawler thread determining the address of a next document to be downloaded by parsing a URL that is set out in a hypertext link, downloading the document corresponding to the URL, and processing the document, as described by Najork, col. 4, lines 59-64.

In the present application, the claimed action sequences are different than the mere selection of hypertext links and parsing of URL's described in the above passage of Najork. The application explains action sequences as follows:

> In one aspect of an embodiment of the crawler of the present invention, the crawler is programmable to perform particular actions sequences for generating queries to the web server. To clarify, consider an example of actions performed by a user to obtain a particular end data set using a conventional web page form.
>
> Referring now to FIG. 3, Web page 140.X is shown in further detail. The page 140.X has two lists 301 and 302, for selecting parameters for generating a query. First list 301 is for selecting a state. Second list 302 is for a profile. In the example, in a conventional web page access, where a user is controlling browser 205, if the user wants to obtain income information for the state of Texas from a Community Facts page in the American Fact Finder web site (factfinder.census.gov), the user performs the following action sequence 305:
>
> 1. Select Texas as the State from list 301.
> 2. Select Income as the Profile from list 302.
> 3. Click on the create button 304, which causes a script 303 to query server 100 with a request that includes the parameters selected from the lists 301 and 302 .
>
> The point to note is that obtaining the desired data requires these actions to be performed, in the proper sequence. Conventional crawlers have not been programmed to do this. Accordingly, in one aspect of an embodiment of the crawler 171 of the present invention, the crawler is programmed as shown at 310 to perform particular actions sequences for generating information 230 to pass to the JEE 210 for generating queries to the web server 100. In the example, the information 230 includes the JavaScript command that invokes script 303 when button 304 is clicked. Also including in the information 230 is a context object, the browser window object, and the document object associated with page 140.X in its context as it exists, loaded in browser 205.

See present application, page 14, line 10 - page 15, line 8.

**DRAFT**

To make the distinction regarding the present invention more clear, claims 4, 12 and 20 are herein amended, as set out above, to state that "the crawler is programmable to perform particular action sequences for selecting non-hypertext-link parameters from the at least one web page in a particular sequence, so that the queries to the web server include the selected parameters and a context arising from the particular sequence." The amendment makes it clear that the claims refer to an action sequence such as that described in the passage set out immediately above, in which non-hypertext-link parameters, such as those in pull-down lists and the like, are selected in a particular sequence. Clearly this is not the sort of sequence taught or suggested by Najork or any of the other cited references.

No new matter is added for these amendments, since the specification as originally submitted provides support, as described in the passage set out immediately above. See also, present application, page 12, line 21 - page 13, line 8 (discussing the importance of context in generating queries).

For the above reasons Applicant contends that claims 4, 12 and 20 are allowable. Further, Applicant contends that the claims are patentably distinct because they respectively depend upon allowable claims.